RL-TR-93-217
In-House Report
December 1993

FAULT COVERAGE MEASUREMENT FOR DIGITAL MICROCIRCUITS

Kevin A. Kwiat, Warren H. Debany, Jr., Heather B. Dussault,
Mark J. Gorniak, Anthony R. Macera, Daniel E. Daskiewich

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-93-217 has been reviewed and is approved for publication.

APPROVED: Eugene C. Blackburn

EUGENE C. BLACKBURN, Chief
Microelectronics & Reliability Division

FOR THE COMMANDER:

HARVEY D. DAHLJELM, Col, USAF
Director of Electromagnetics & Reliability

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (ERDA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	In-House Oct 88 - Sep 93

4. TITLE AND SUBTITLE
FAULT COVERAGE MEASUREMENT FOR DIGITAL MICROCIRCUITS

5. FUNDING NUMBERS
PE - 62702F
PR - 2338
TA - 01
WU - 7C

6. AUTHOR(S)
Kevin A. Kwiat, Warren H. Debany, Jr., Heather B. Dussault, Mark J. Gorniak, Anthony R. Macera, Daniel E. Daskiewich

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Rome Laboratory (ERDA)
525 Brooks Rd
Griffiss AFB NY 13441-4514

8. PERFORMING ORGANIZATION REPORT NUMBER
RL-TR-93-217

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (ERDA)
525 Brooks Rd
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Kevin A. Kwiat/ERDA (315) 330-2047

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Procedure 5012 governs the measurement and reporting of fault coverage for digital microcircuits. This report expands on the published version of Procedure 5012 and explains the rationale behind the requirements of the procedure. The complete text of 5012, with additional annotations, is included in this technical report.

14. SUBJECT TERMS
digital microcircuit, fault coverage, fault detection
fault simulation, testing

15. NUMBER OF PAGES
32

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	U/L

INTRODUCTION

Procedure 5012 governs the measurement and reporting of fault coverage for digital microcircuits. This procedure was first published as part of MIL-STD-883 Notice 11 (18 Dec 1989) and was revised in Notice 12 (27 Jul 1990). This technical report expands on the published version of Procedure 5012 and explains the rationale behind the procedure's requirements. The complete text of 5012, with additional annotations, is included in this technical report. There are some editorial differences between the text given in this technical report and the published version of 5012. In the event that a significant difference exists between the two documents (such as one that affects meaning, or where a verbatim quote is required), the published version of 5012 shall govern.

The motivation to develop a standardized procedure for fault coverage measurement arises because of the need to improve microcircuit quality levels in electronic equipment. The IC field reject rate, or "outgoing quality level," is the fraction of ICs that pass all tests at manufacturing-level test yet are faulty. Recent directives [2] require a field reject rate (after environmental stress screening) of no more than 100 parts per million (ppm) or 0.01%. It has been shown (for example, by Wadsack [3]) that there is a relationship between the fault coverage of the manufacturing tests, the measured test yield (fraction of ICs that pass the manufacturing tests), and the field reject rate due to logic faults alone. Let

- f denote the fault coverage of a test vector sequence (expressed as a fraction, e.g., 0.95 for 95%)
- m denote the measured test yield (i.e., fraction of ICs that pass the test vector sequence)
- r denote the field reject rate (i.e., fraction of devices that pass the test vector sequence yet are faulty)

Then

$$r = \frac{(1 - f)(1 - m)}{1 - (1 - f)m}$$

Very little information has been made publicly available concerning actual IC yields and field reject rates. A study has been documented [4][5] that examined the consequences of testing a microprocessor, the MC6802, with a test vector set with 96.6% fault coverage, versus testing using a test vector set with 99.9% fault coverage. The field reject rate estimated by the authors of the MC6802 study, obtained by determining the number of ICs that passed at 96.6% fault coverage but failed at 99.9% fault coverage, equated to 8,200 ppm. The measured test yield at 96.6% fault coverage was 70.7%; using Wadsack's model the predicted field reject rate is 10,200 ppm (which closely matches that estimated in the MC6802 study).

Even when rescreening of ICs is performed by the customer who receives them, testing usually consists only of checking that electrical and switching performance are within specifications, and if logic testing is performed then *at best* all that is done is to apply the same test (with the same fault coverage) that was originally applied by the manufacturer. To show what is implied by a high field reject rate, consider a circuit board assembled using 50 ICs where each IC type used has an outgoing quality level of 10,200 ppm. With just over 1% of the ICs on average being faulty, the probability that such a board initially would have

only fault-free ICs is only 60%. For lower fault coverage the effects are more drastic; at a fault coverage level of 90% the measured test yield would have been about 72.1% and the outgoing quality level would have been 30,000 ppm, resulting in a probability of 21.8% that the board would contain 50 fault-free ICs. Clearly, manufacturing-level tests for ICs must have high fault coverage in order to reduce costly board (and higher-level) *test generation*, *testing*, and *rework* in order to eliminate faulty components.

It has long been known that different fault simulators commonly produce drastically different results for identical logic models and test vector sets. The Radiation-Hardened 32-Bit (RH32) Processor program at Rome Laboratory provided the original motivation to develop a standardized method for measuring fault coverage consistently, where it was expected that fault simulation would be performed using a variety of fault simulators. Later, the annex of the RH32 Statement of Work that concerned fault coverage measurement was used as the basis of a requirements section in the draft implementation plan for the VHSIC/VLSI Qualification Procedures (the "Qualified Manufacturers List" or "QML") program.

Under an Expert Science and Engineering task with the University of South Florida, experiments were performed with four commercially-available fault simulators in order to identify what differences are possible, why they occur, and what can be done by using modeling guidelines, simulation directives, and postprocessing in order to reduce or eliminate differences in reported fault coverage. MIL-I-38535 for QML now references 5012, as do the MIL-M-38510 detail specifications for gate arrays. In 1987, the requirements now detailed in 5012 were made part of Requirement 64 of MIL-STD-454L:

> 4.5.2 *Fault coverage.* Fault coverage shall be reported for the manufacturing-level logic tests for all digital microcircuits designed after 30 September 1988. Fault coverage shall be based on the equivalence classes of single, permanent, stuck-at-zero and stuck-at-one faults on all lines of a TISSS-compatible structural VHDL model, where the structural model is expressed in terms of gate-level primitives or simple atomic functions (such as flip-flops). Large, regular structures such as RAMs and ROMs shall not be modeled at the gate level, but rather documentation shall be provided that these structures are tested using appropriate algorithms (such as galloping patterns for a RAM).

The draft of MIL-STD-454 that is being circulated at the time of this writing has been revised to simply reference MIL-STD-883 Procedure 5012.

The authors wish to acknowledge the help received from all of the people and organizations who reviewed drafts of 5012 and provided comments. While the overwhelming response was positive, the authors were particularly gratified to note that reviewers, without exception, were honest and specific in their criticisms. As always, the most useful comments were the negative ones, and the authors have tried where possible either to accommodate or at least to answer every issue that was raised.

John J. Bart, Chief Scientist of Reliability Sciences, provided the opportunity to develop 5012 and made numerous suggestions that contributed to its acceptability. Dr. Sami Al-Arian, of the University of South Florida, under contract to Rome Laboratory performed much of the technical work that resulted in usable techniques for reducing differences between fault simulators. Charles G. Messenger, Chief of the Reliability and Diagnostics Branch,

provided greatly-appreciated assistance in developing, circulating, testing, and revising 5012. The authors wish to thank John P. Farrell, formerly chief of the Reliability Assurance Branch of RADC, whose initial idea it was to convert an annex in the RH32 Statement of Work into a standard test procedure, and who quietly made a number of suggestions that got the development of 5012 over a number of rough spots.

References

[1] Debany, W.H., K.A. Kwiat, H.B. Dussault, M.J. Gorniak, A.R. Macera, and D.E. Daskiewich, "Fault coverage measurement for digital microc rcuits," MIL-STD-883 Test Procedure 5012, Rome Air Development Center (RBRA), Griffiss AFB NY 13441, 18 December 1989 (Notice 11) and 27 July 1990 (Notice 12).

[2] "Air Force Policy Letter #1 on R&M 2000: Environmental St ess Screening, General Guidelines and Minimum Requirements," Department of the Air Force, Office of the Chief of Staff, December 1985.

[3] Wadsack, R.L., "Fault Coverage in Digital Integrated Circuits," *Bell System Technical Journal*, May/June 1978, pp 1475-1488.

[4] Harrison, R.A., R.W. Holzwarth, R.R. Motz, R.G. Daniels, J.S. Thomas, and W.H. Weimann, "Logic Fault Verification of LSI: How It Benefits the User," *Proceedings, WESCON*, 1980, paper 34/1.

[5] Daniels, R.G. and W.C. Bruce, "Built-In Self-Test Trends in Motorola Microprocessors," *IEEE Design & Test of Computers*, April 1985, pp 64-71.

[6] Al-Arian, S.A. and K.A. Kwiat, "Defining a Standard for Fault Simulator Evaluation," *Proceedings, International Test Conference*, 1988, p. 1001.

[7] Al-Arian, S.A., M. Nordenso, H. Kunmenini, H. Abujbara, and J. Wang, "Fault Simulator Evaluation," RADC-TR-89-230, November 1989.

[8] Levi, M.W., "CMOS is most testable," *Proceedings, IEEE International Test Conference*, 1981, pp 217-220.

[9] Fritzemeier, R.R., J.M. Soden, R.K. Treece, and C.F. Hawkins, "Increased CMOS IC stuck-at fault coverage with reduced I_{DDQ} test sets," *Proceedings, International Test Conference*, 1990, pp 427-434.

[10] Carroll, M., "Built-in array payoff: better fault detection," *High Performance Systems*, August 1989, pp 28-44.

OUTLINE OF PROCEDURE 5012

1. PURPOSE
1.1 Terms

2. APPARATUS
2.1 Logic Simulator
2.2 Fault Simulator

3. PROCEDURE
3.1 Logic Model
   3.1.1 Level of Modeling
   3.1.2 Logic Lines and Nodes
   3.1.3 Gate-Logic (G-logic) and Block-Logic (B-Logic) Partitions
   3.1.4 Model Hierarchy
   3.1.5 Fractions of Transistors
3.2 Fault Model
   3.2.1 G-Logic
   3.2.2 B-Logic
      3.2.2.1 Built-In Self-Test
3.3 Fault Universe Selection and Fault Equivalence Classing
   3.3.1 Initial Fault Universe
   3.3.2 Fault Equivalence Classes
   3.3.3 Detectable Fault Universe
3.4 Fault Simulation
   3.4.1 Automatic Test Equipment Limitations
   3.4.2 G-Logic
      3.4.2.1 Hard Detections and Potential Detections
      3.4.2.2 Fault Simulation Procedures
         3.4.2.2.1 Procedure 1: Full fault simulation
         3.4.2.2.2 Procedure 2: Obtain lower bound on actual fault coverage using fixed sample size
         3.4.2.2.3 Procedure 3: Accept/reject lower bound on actual fault coverage using fixed sample size
   3.4.3 B-Logic
3.5 Fault Coverage Calculation

4. SUMMARY

TEXT OF
MIL-STD-883
PROCEDURE 5012
FAULT COVERAGE MEASUREMENT
FOR DIGITAL MICROCIRCUITS

1. *PURPOSE.* This test procedure specifies the methods by which fault coverage is reported for a test program applied to a microcircuit herein referred to as the Device Under Test (DUT). This procedure describes requirements governing the development of the logic model of the DUT, the assumed fault model and fault universe, fault classing, fault simulation, and fault coverage reporting. This procedure provides a consistent means of reporting fault coverage regardless of the specific logic and fault simulator used. Three procedures for fault simulation are described in this procedure: full fault simulation and two fault sampling procedures. The applicable procurement document shall specify a minimum required level of fault coverage and, optionally, specify the procedure to be used to determine the fault coverage. A Fault Simulation Report shall be provided that states the fault coverage obtained, as well as documenting assumptions, approximations, and procedures used.

Where any technique detailed in this procedure is inapplicable to some aspect of the logic model, or inconsistent with the functionality of the available fault simulator and simulation postprocessing tools, it is sufficient that the user of this procedure employ an equivalent or comparable technique and note the discrepancy in the fault simulation report.

Microcircuits may be tested by nontraditional methods of control or observation, such as power supply current monitoring or the addition of test points that are available by means of special test modes. Fault coverage based on such techniques shall be considered valid if substantiating analyses or references are provided in the fault simulation report.

> *NOTE:* This test procedure deals with microcircuit *quality*, not *reliability*. It does not attempt to relate logic model fault coverage to microcircuit failure rates; in fact, there is not necessarily any direct relationship between quality and reliability. However, mathematical models have been developed that relate fault coverage and test yield to "quality levels" or "field reject rates."
>
> This test procedure deals only with the *means* of fault simulation. It does not set specific requirements or goals for minimum required levels of fault coverage. This procedure does not recommend either for or against the use of statistical fault sampling techniques. *(End of Note)*

1.1 *Terms.* Terms and abbreviations not defined elsewhere in the text of this test procedure are defined in this section.

a. *Automatic Test Equipment (ATE).* The apparatus with which the actual DUT will be tested. ATE includes the ability to apply a test vector sequence (see 1.11, *Test vector sequence*).

b. *Broadside application.* A method of applying a test vector sequence where input stimuli change only at the beginning of a simulation cycle or ATE cycle and all changes on primary inputs of the DUT are assumed to be simultaneous. Non-broadside application occurs when test vectors are conditioned by additional timing information such as delay (with respect to other primary inputs), return-to-zero, return-to-one, and surround-by-complement.

c. *Detection.* An error at an observable primary output of a logic model caused by the existence of a logic fault. A *hard detection* is where an observable output value in the fault-free logic model is distinctly different from the corresponding output value in the faulty logic model. An example of a hard detection is where the fault-free logic model's output value is 0 and the faulty logic model's output value is 1, or where the fault-free logic model's output value is 1 and the faulty logic model's output value is 0. If the high-impedance state (Z) can be sensed by the ATE, then a hard detection can involve the Z state as well. A *potential detection* is an error where the fault-free output is 0 or 1 and the faulty output value is unknown (X), or Z if Z cannot be sensed by the ATE.

> *NOTE:* In the literature a "potential detection" is also referred to as a "possible detection."
>
> The Z state can be sensed by the ATE when active or passive loads on the DUT's outputs are used. Using passive loads, Z states can be tested in two passes: in the first pass, the high-impedance outputs are pulled up and each expected Z response in the output is converted to a 1; in the second pass, the high-impedance outputs are pulled down and each expected Z response in the output is converted to a 0. *(End of Note)*

d. *Established test algorithm.* An algorithm, procedure, or test vector sequence, that when applied to a logic component or logic partition has a known fault coverage or test effectiveness. This fault coverage or test effectiveness is denoted herein as the *established fault coverage* or *established test effectiveness* for the established test algorithm. For example, an established test algorithm for a RAM may be a published memory test algorithm, such as GALPAT, that has been shown by experience to detect essentially "all" RAM failures and therefore is assessed an established test effectiveness of 100%. An ALU may be tested by means of a precomputed test vector sequence for which fault coverage has been previously determined. More than one established test algorithm may exist for a logic component or logic partition, each with a different established fault coverage or test effectiveness.

> *NOTE:* For example, some memory test algorithms detect only stuck-at faults and some decoder faults, but not coupling or pattern-sensitivity faults. Documentation of the established test effectiveness must reflect the limitations of the fault models that are covered by such algorithms. *(End of Note)*

e. *Failure hierarchy: failure mechanism, physical failure, logical fault, error.* The failure hierarchy relates physical defects and their causes to fault simulators and observable

effects. A failure mechanism is the actual cause of physical failure; an example is electromigration of aluminum in a microcircuit. A *physical failure* (or simply *failure*) is the actual physical defect caused by a failure mechanism; an example is an open metal line. A *logical fault* (or simply *fault*) is a logical abstraction of the immediate effect of a failure; an example is "stuck-at-one" behavior of a logic gate input in the presence of an open metal line. An *error* is a difference between the behavior of a fault-free and faulty DUT at one or more observable primary outputs of the DUT.

f. *Fault coverage.* For a logic model of a DUT, a fault universe for the logic model of the DUT, and a given test vector sequence, fault coverage is the fraction obtained by dividing the number of faults contained in the fault universe that are detected by the test vector sequence by the total number of faults contained in the fault universe. Fault coverage is also stated as a percentage. In this test procedure fault coverage is understood to be based on the *detectable fault equivalence classes* (see 3.3). Rounding of fault coverage fractions or percentages shall be "toward zero," not "to nearest." For example, if 9,499 faults are detected out of 10,000 faults simulated, the fault coverage is 94.99%; if this value is to be rounded to two significant digits, the result shall be reported as 94%, not 95%.

> *NOTE:* Using truncation instead of rounding avoids the possibility of having a "noncompliant" level of fault coverage rounded up to a "compliant" level. *(End of Note)*

g. *Logic lines, Nodes. Logic lines* are the connections between components in a logic model, through which logic signals flow, are *logic lines*. Logic lines are the idealized "wires" in a logic model. A set of connected logic lines is a *node*.

> *NOTE:* In the literature a logic line is also referred to as a *signal line*, or *wire*. A node is also referred to as a *net*. *(End of Note)*

h. *Combinational and Sequential logic.* Combinational digital logic contains only components that do not possess memory, and in which there are no feedback paths. Sequential digital logic contains at least one component that contains memory, or at least one feedback path, or both. For example, a flip-flop is a component that contains memory, and cross-coupled logic gates introduce feedback paths.

i. *Macro.* A logic modeling convention representing a model contained within another model. A macro boundary does not necessarily imply the existence of a physical boundary in the logic model. A "main model" is a logic model that is not contained within a larger model. Macros may be nested (that is, a macro may contain submacros).

> *NOTE:* In the literature a macro is also referred to as a *submodel. (End of Note)*

j. *Primary inputs, Primary outputs.* Primary inputs to a logic model represent the logic lines of a DUT that are driven by the ATE's drivers and thus are directly controllable

test points. Primary outputs from a logic model represent the logic lines of the DUT that are sensed by the ATE's comparators and thus are directly observable test points. The inputs to the "main model" of the logic model of the DUT are the *primary inputs*, and the outputs from the main model are the *primary outputs*. Internal nodes that can be driven or sensed by means of special test modes shall be considered to be control or observation test points.

NOTE: Some test techniques and design-for-testability approaches may provide additional control test points or observation test points that, for the purpose of this procedure, can be used during fault simulation in order to contribute to fault detection. For example, power supply current monitoring for CMOS circuits (I_{DDQ}) permits the power supply lines to be considered observable test points under some conditions. *(End of Note)*

k. *Test effectiveness.* A measure similar to fault coverage, but used in lieu of fault coverage in cases where physical failures cannot be modeled accurately as logical faults. For example, many RAM and PLA failures cannot be idealized conveniently in the same way as gate-level failures. However, established test algorithms may be used to detect essentially all likely physical failures in such structures.

l. *Test vector sequence.* The (ordered) sequence of stimuli (applied to a logic model of a DUT) or stimulus/response values (applied to, and compared for, the actual DUT by the ATE).

m. *Undetectable and Detectable faults.* An *undetectable fault* is defined herein as a logical fault for which no test vector sequence exists that can cause at least one hard detection or potential detection (see 1.1c, *Detection*). Otherwise (that is, some test vector sequence exists that causes at least one hard detection, or potential detection, or both), the fault is defined herein to be a *detectable fault* (see 3.3.3).

NOTE: By this definition, it is sufficient for a fault to cause a potential detection for the fault to be declared detectable. However, credit is not given for potential detections in determining fault coverage unless it is shown that the potential detection *implies* hard detection (see 3.4.2.1). *(End of Note)*

2. *APPARATUS.*

2.1 *Logic Simulator.* Implementation of this test procedure requires the use of a facility capable of simulating the behavior of fault-free digital logic in response to a test vector sequence; this capability is herein referred to as *logic simulation*

In order to simulate sequential digital logic, the simulator must support simulation of a minimum of four logic states: zero (0), one (1), high-impedance (Z), and unknown (X). In order to simulate combinational digital logic only, the simulator must support simulation of a minimum of two logic states: 0 and 1.

At the start of logic simulation of a logic model of a DUT containing sequential logic, the state of every logic line and component containing memory shall be X; any other initial condition, including explicit initialization of any line or memory element to 0 or 1, shall be documented and justified in the Fault Simulation Report.

In order to simulate "wired connections" or "bus" structures the simulator must be capable of resolving signal conflicts introduced by such structures. Otherwise, modeling workarounds shall be permitted to eliminate such structures from the logic model (see 3.1.2).

In order to simulate sequential digital logic, the simulator must support event-directed simulation. As a minimum, unit-delay logic components must be supported.

Simulation of combinational-only logic, or simulation of sequential logic in special cases (such as combinational logic extracted from a scannable sequential logic model) can be based on non-event-directed simulation, such as levelized, zero-delay, or compiled-code methods. The Fault Simulation Report shall describe why the selected method is equivalent to the more general event-directed method.

2.2 *Fault Simulator.* In addition to the capability to simulate the fault-free digital logic, the capability is also required to simulate the effect of single, permanent, stuck-at-zero and stuck-at-one faults on the behavior of the logic; this capability is herein referred to as *fault simulation.* Fault simulation shall reflect the limitations of the target ATE (see 3.4.1). It is not necessary that the fault simulator directly support the requirements of this test procedure in the areas of hard vs. potential detections, fault universe selection, and fault classing. However, the capability must exist, at least indirectly, to report fault coverage in accordance with this procedure. Where approximations are used (for example, where fault classing compensates for a different method of fault universe selection) such differences shall be documented in the Fault Simulation Report, and it shall be shown that the approximations do not increase the fault coverage obtained.

> *NOTE:* This test procedure places requirements on how the logic model for a DUT is developed for use with a fault simulator. Postprocessing of a fault simulator's output may be necessary in order to report fault coverage in a manner consistent with this procedure. *(End of Note)*

3. *PROCEDURE.*

3.1 *Logic Model.*

3.1.1 *Level of Modeling.* The DUT shall be described in terms of a logic model composed of components and connections between components. Primary inputs to the logic model are assumed to be outputs of an imaginary component (representing the ATE's drivers), and primary outputs of the logic model are assumed to be inputs to an imaginary component (representing the ATE's comparators). Some logic simulators require that the ATE drivers and comparators be modeled explicitly; however, these components shall not be considered to be part of the logic model of the DUT.

NOTE: In the literature, a logic model is also referred to as a *netlist. (End of Note)*

3.1.2 *Logic Lines and Nodes.* (See 1.1g, *Logic lines, Nodes.*) All fanout from a node in a logic model is ideal; that is, *fanout branches* associated with a node emanate from a single point driven by a *fanout origin.* All fanin to a node in a logic model is ideal; that is, multiple *fanin branches* in a node drive a single line. Figure 1 shows a node that includes fanin branches, a fanout origin, and fanout branches. Because fanin and fanout generally are not ideal in actual circuit layout, the actual topology of the circuit should be modeled, if it is known, by appropriately adding single-input non-inverting buffers to the logic model.

Modeling workarounds may be used to eliminate fanin to a node. This may be required if the simulator does not directly model "wired connections" or "bus" structures. Some simulators may permit internal fanin, but require that bidirectional pins to a DUT be modeled as separate input and output functions.

NOTE: In the literature, a fanout origin is also referred to as a *fanout stem. (End of Note)*

3.1.3 *G-Logic and B-Logic Partitions.* Simple components of the logic model (logic primitives such as AND, OR, NAND, NOR, XOR, buffers, or flip-flops; generally the indivisible primitives understood by a simulator) are herein referred to as *gate logic* (*G-logic*). Complex components of the logic model (such as RAM, ROM, or PLA "primitive" components, and behavioral models – relatively complex functions that are treated as "black boxes" for the purpose of fault simulation) are referred to herein as *block logic* (*B-logic*).

For the purpose of fault simulation, the logic model shall be divided into non-overlapping logic partitions; however, the entire logic model may consist of a single logic partition. The logic partitions contain components and their associated lines; although lines may span partitions, no component is contained in more than one partition. A G-logic partition contains only G-logic; any other logic partition is a B-logic partition.

A logic partition consisting of G-logic, or B-logic, or G-logic and B-logic that, as a unit, is testable using an established testing algorithm, with known fault coverage or test effectiveness, may be treated as a single B-logic partition.

NOTE: The interconnection of B-logic components, with G-logic "glue," can form a B-logic partition. For example, a "64Kx8 RAM" in a logic model may actually be composed of 32 16Kx1 RAM primitives and decoding logic. However, a GALPAT algorithm that exploits the 16Kx1 organization of the memory would be more efficient than one that treats the 64Kx8 structure as a single component.

Although fault simulation can be performed at the transistor-primitive level, such simulation is discouraged for two reasons. First, fault simulation at the transistor level is far more time-consuming than at the "gate" level. Second, transistor-level fault simulation generates a large fraction of potential detections that are difficult to justify as legitimate hard detections, which significantly reduces the accuracy of fault simulation.

A G-logic partition may contain disjoint or unconnected logic. In particular, G-logic comprising the logic lines associated with B-logic appear to be unconnected with the primary inputs and/or primary outputs. However, this partitioning of logic is strictly for the purpose of fault selection. It is not intended that the partitioning of logic force the simulation of logic in separate passes, although it may facilitate breaking a large fault simulation into multiple smaller passes. Logic models shall be fault-simulated explicitly only with respect to the "gate-level" primitives. "Block-level" primitives generally are represented by behavioral models, complex primitives, or "indivisible" submodels, where fault effects are merely propagated through such structures. *(End of Note)*

3.1.4 *Model Hierarchy.* The logic model may be hierarchical (that is, consisting of macro building blocks), or flat (that is, a single level of hierarchy with no macro building blocks). Hierarchy does not impose structures on lines; for example, there is no implied fanout origin at a macro input or output. Macros that correspond to physical partitions in a model shall use additional buffers (or an equivalent method) to enforce adherence to the actual DUT's fanout.

3.1.5 *Fractions of Transistors.* The fraction of transistors comprising each G-logic and B-logic partition, with respect to the total count of transistors in the DUT, shall be determined or closely estimated; the total sum of the transistor fractions shall equal 1. Where the actual transistor counts are not available, estimates may be made on the basis of gate counts or microcircuit area; the assumptions and calculations supporting such estimates shall be documented in the Fault Simulation Report. The transistor fractions shall be used in order to weight the fault coverage measured for each individual logic partition (see 3.5).

NOTE: Fault coverage is weighted by transistor fractions for the following reasons:

a. Transistor counts are available early in the design process and are insensitive to variations in placement and routing. Transistor counts can be estimated closely from gate counts or obtained exactly from circuit-level CAD data. Other physical circuit characteristics (such as area and interconnect lengths) are unknown before the design is essentially complete and are quite variable.

b. Failure rates generally are unknown for internal microcircuit structures, and most failure rate models are based on transistor counts in the first place. In any case, this procedure does not assume that any relationship exists between fault coverage and reliability.

(End of Note)

3.2 *Fault Model.*

3.2.1 *G-Logic.* The fault model for G-logic shall be permanent stuck-at-zero and stuck-at-one faults on logic lines. Only single stuck-at faults are considered in calculating fault coverage.

NOTE: Obviously, many other types of logical faults (such as bridging, pattern-sensitive, transient, and multiple faults) should be considered in order to accurately model commonly-occurring types of physical failures. However, practical limitations of the currently-available fault simulation tools make it unreasonable to require the determination of fault coverage based on other than single, permanent, stuck-at-zero and stuck-at-one faults. *(End of Note)*

3.2.2 *B-Logic.* No explicit fault model is assumed for B-logic components. However, an established test algorithm shall be applied to each B-logic component or logic partition. If a B-logic partition contains logic lines and/or G-logic components, justification shall be provided in the Fault Simulation Report as to how the established test algorithm that is applied to the B-logic partition detects faults associated with the logic lines and G-logic components.

NOTE: For example, an embedded RAM block, for which no information is available concerning its implementation, may be tested using a full GALPAT test (a time-consuming algorithm that is generally considered to have a "test effectiveness" of 100%). If information such as physical partitioning or a bit map is available for a RAM, then the RAM testing may be shortened by using a more efficient algorithm. *(End of Note)*

3.2.2.1 *Built-In Self-Test.* A special case of B-logic is a B-logic partition that includes a linear-feedback shift register (LFSR) that performs "signature analysis" for compression of output error data. Table I lists penalty values for different LFSR degrees. If the LFSR implements a *primitive* GF(2) polynomial of degree k, where there is at least one flip-flop stage between inputs to a multiple-input LFSR, then the following procedure shall be used in order to determine a lower bound on the established fault coverage of the logic partition:

Step 1: Excluding the LFSR, but including any stimulus generation logic considered to be part of the logic partition, determine the fault coverage of the logic partition by fault simulation without signature analysis; denote this fault coverage by C.

Step 2: Reference table I. For a given degree k obtain the penalty value p. The established fault coverage of the logic partition using a LFSR of degree k shall be reported as $(1 - p)C$. That is, a penalty of $(100p)\%$ is incurred in assessing the effectiveness of signature analysis if the actual effectiveness is not determined.

NOTE: The penalty values listed in table I are based on preliminary work in-house at RADC. Experiments were run that determined confidence intervals on error escape, for actual logic circuits, for different polynomial types and degrees. *(End of Note)*

3.3 *Fault Universe Selection and Fault Equivalence Classing.* Fault coverage shall be reported in terms of equivalence classes of the detectable faults. This section describes the selection of the initial fault universe, the partitioning or collapsing of the initial fault universe into fault equivalence classes, and the removal of undetectable faults in order to form the detectable fault universe. These three stages constitute the fault simulation reporting requirements; however, it is generally more efficient to obtain the set of faults that represent the fault equivalence classes directly without explicitly generating the initial fault universe.

> *NOTE:* Equivalence classes are used as the basis for calculating fault coverage, instead of the (uncollapsed) set of "all" faults, for several reasons. First, there is no reason to believe that the set of equivalence classes yields results that are less *accurate* than those obtained by using the set of "all" faults, in the sense that the set of "all" faults would relate more closely to physical failures. Second, just as with any other laboratory instrument, the fault simulator should be *precise*, meaning that its results are repeatable with itself and with other instruments that perform the same function.
>
> Addressing the issue of accuracy, one might be tempted to say that fault coverage based on equivalence classes is skewed because a class that contains, say 10 faults, weights each fault with only $1/10^{th}$ of its "importance." But how is "importance" measured? The set of "all" faults does not reweigh faults according to physical characteristics such as length of logic lines, proximity to other lines, etc.
>
> Addressing the issue of precision, the repeatability of fault coverage results is greatly affected by factors such as level of modeling, handling of unknown values, and so on. A study sponsored by RADC (see [6] and [7]) documented the drastic differences obtained from four fault simulation packages that were tested on identical sets of logic models and test vector sequences. It was determined that both the simplest and most effective way to obtain consistent results among the simulators was to have each fault simulator report fault coverage in terms of *its own* fault equivalence classes; the differences in both *fault selection* and *fault classing* strategies appeared to cancel out so as to yield consistent results among all four simulators. *(End of Note)*

3.3.1 *Initial Fault Universe.* The initial fault universe shall consist of single, permanent, stuck-at-zero and stuck-at-one faults on every logic line (not simply on every logic node) in the G-logic partitions of the logic model.

A bus, which is a node with multiple driving lines, shall be considered, for the purpose of fault universe generation, to be a multiple-input, single-output logic gate. The initial fault universe shall include stuck-at-zero and stuck-at-one faults on each fanin and fanout branch and the fanout origin of the bus (see figure 1).

The fault universe does not explicitly contain any faults within B-logic partitions. However, all faults associated with inputs and outputs of B-logic components either are contained in a G-logic partition or shall be shown to be considered by established test algorithms that are applied to the B-logic partitions.

No faults shall be added or removed by considering or not considering logic model hierarchy. No extra faults shall be associated with any primary input or output line, macro input or output line, or logic line that spans logic partitions where the logic partitions do not correspond to a physical boundary.

No more than one stuck-at-zero and one stuck-at-one fault per logic line shall be contained in the initial fault universe.

3.3.2 *Fault Equivalence Classes.* The initial fault universe shall be partitioned or collapsed into "fault equivalence classes" for reporting purposes. The fault equivalence classes shall be chosen such that all faults in a fault equivalence class cause apparently identical erroneous behavior with respect to the observable outputs of the logic model. One fault from each fault equivalence class shall be selected to represent the fault class for reporting purposes; these faults shall be called the *representative faults.*

For the purpose of implementing this test procedure it is sufficient to apply simple rules to identify structurally-dependent equivalence classes. An acceptable method for selecting the representative faults for the initial fault universe consists of listing all single, permanent, stuck-at faults as specified in table II. Any other fault equivalencing procedure used shall be documented in the Fault Simulation Report.

If a bus node exhibits wired-AND or wired-OR behavior in the applicable circuit technology, then faults associated with that bus shall be collapsed in accordance with the AND or OR fault equivalencing rules, respectively. Otherwise, no collapsing of faults associated with a bus shall be performed.

3.3.3 *Detectable Fault Universe.* Fault coverage shall be based on the *detectable fault universe.* Undetectable faults shall be permitted to be dropped from the set of representative faults; the remaining set of representative faults comprises the detectable fault universe. In order for a fault to be declared as undetectable, documentation shall be provided in the Fault Simulation Report as to why there does not exist any test vector sequence capable of guaranteeing that the fault will cause an error at an observable primary output (see 1.1m, *Undetectable and Detectable faults*). Any fault not documented in the Fault Simulation Report as being undetectable shall be considered detectable for the purpose of calculating fault coverage.

> *NOTE:* In general, identifying undetectable faults (in order to obtain the detectable fault universe) is a difficult problem. However, undetectable faults associated with some simple structural dependencies can be easily identified. Chiefly, these are in four areas:
>
> a. Logic with no path to a primary output of the logic model. For example, an unused output from a flip-flop has no path to an observable output and so both stuck-at faults associated with the unused output are undetectable.
>
> b. Stuck-at faults associated with locked-at values. For example, a gate input that is connected to ground always has the state 0; therefore, stuck-at-zero on this line is undetectable.

c. Faults associated with deliberately designed logical redundancies. For example, redundancy may be introduced for additional drive capability, locked-out options in logic, or the addition of "bridging" terms in a Boolean function to avoid hazards.

d. Faults that are contained within small logic partitions that can be exhaustively fault-simulated. For example, a common realization of an arithmetic/logic unit (ALU), involves constructing the ALU out of four-bit slices. If each four-bit slice has a limited number of inputs (generally twelve to fourteen), then a single four-bit slice can be extracted from the logic model and fault simulation can be performed exhaustively on that slice to obtain a list of faults associated with that slice that are undetectable even when the inputs and outputs of that slice are directly accessible. Therefore, those faults certainly are undetectable when that slice is embedded within any containing logic model.

(End of Note)

3.4 *Fault Simulation.*

3.4.1 *Automatic Test Equipment Limitations.* Fault coverage reported for the logic model of a DUT shall reflect the limitations of the target ATE. Two common cases are:

a. Fault detection during fault simulation shall occur only at times where the ATE will be capable of sensing the primary outputs of the DUT; there must be a one-to-one correspondence between simulator compares and ATE compares. For example, if fault coverage for a test vector sequence is obtained using broadside fault simulation (where fault detection occurs after every change of input stimuli, including clock signals), then it is not correct to claim the same fault coverage on the ATE if the test vectors are reformatted into cycles where a clock signal is pulsed during each cycle and compares occur only at the end of each cycle.

b. If the ATE cannot sense the Z output state (either directly or by multiple passes), then the reported fault coverage shall not include detections involving the Z state. That is, an output value of Z shall be considered to be equivalent to an output value of X.

Any differences in format or timing of the test vector sequence, between that used by the fault simulator and that applied by the ATE, shall be documented in the Fault Simulation Report and it shall be shown that fault coverage achieved on the ATE is not lower than the reported fault coverage.

3.4.2 *G-Logic.*

3.4.2.1 *Hard Detections and Potential Detections.* Fault coverage for G-logic shall include only faults detected by hard detections. Potential detections shall not be considered directly

in calculating the fault coverage. No number of potential detections of a fault shall imply that the fault would be detected.

Some potential detections can be converted into hard detections for the purpose of calculating fault coverage. If it can be shown that a fault is only potentially detected by fault simulation but is in fact detectable by the ATE by a difference not involving an X value, then upon documenting those conditions in the Fault Simulation Report that fault shall be considered to be detected as a hard detection and the fault coverage shall be adjusted accordingly.

> *NOTE:* Clock line faults provide a common example of where a fault may be detectable on one or the other of two test vectors, but not both vectors. For example, consider a
>
> D-flip-flop with two inputs: data and clock. Both stuck-at-zero and stuck-at-one on the clock input are detectable by the fault simulator only as potential detections. However, if both stuck-at-zero and stuck-at-one faults on the data input are shown to be detected as hard detections by the fault simulator, then, regardless of the initial state of the flip-flop, it is guaranteed that the ATE would detect an error at some point in the test vector sequence if either stuck-at fault were to exist on the clock line. *(End of Note)*

Faults associated with three-state buffer enable signal lines can cause X states to occur on nodes with fanin branches, or erroneous Z states to occur on three-state primary outputs that may be untestable on some ATE. These faults may then be detectable only as potential detections, but may be unconvertible into hard detections. In such cases, it is permissible for the Fault Simulation Report to state separately the fraction of the undetected faults that are due to such faults.

3.4.2.2 *Fault Simulation Procedures.* The preferred method of fault simulation for G-logic is to simulate the effect of each representative fault in the G-logic. However, this may not be practical in some cases due to the large number of representative faults, or because of limitations of the logic models or simulation tools. In such cases fault sampling procedures may be used. When fault sampling is used, either the procurement document shall specify the method of obtaining a "random" sample of faults or the Fault Simulation Report shall describe the method used. In either case, the complete random sample of faults shall be obtained before beginning the fault simulation procedure involving a random sample of faults.

> *NOTE:* The procurement document that permits the use of statistical fault sampling must address several problems:
>
> a. How is the "random" sample of faults to be chosen? The most "fair" approach is to prepare a list of every possible fault (that is, the full set of representatives of the fault equivalence classes) and use a random number generator to select the subset of faults to be simulated. In any case, it is not proper to use a method that skews the "distribution" of faults, such

as faulting only the signal lines accessible at the main model level without faulting within submacros.

b. When re-fault-simulating a test vector sequence, after reorganizing, revising, or adding to the sequence, should the same subset of faults be simulated, or should a "fresh" sample of faults be obtained? Both approaches have advantages and disadvantages (to the customer). If the same subset is simulated, there is the opportunity (by design or by accident) to target the detection of specific undetected faults by the new test vector sequence. Thus, a high fault coverage may be obtained by the fault sampling procedure yet the actual fault coverage may not have been significantly improved. However, if a fresh sample is used for each fault simulation pass, then with each new pass there is a probability (here designed to be $\leq$ 5%) that a fault coverage that is higher than the actual fault coverage could be reported. Thus, simply resimulating the same test vector sequence repeatedly with new random subsets of faults can result in an erroneous lower bound on fault coverage.

(End of Note)

Use of any fault simulation procedure other than Fault Simulation Procedure 1 (see 3.4.2.2.1) shall be documented and justified in the Fault Simulation Report.

In this section, it is assumed that the representative faults declared to be undetectable have been removed from the set of faults to be simulated.

NOTE: This test procedure does not specify the type of fault simulation that is performed. It is equally acceptable to fault simulate by simulating the effect of each fault individually on the behavior of the logic model of the DUT (serial fault simulation) or to use more sophisticated methods such as parallel, deductive, or concurrent fault simulation. *(End of Note)*

3.4.2.2.1 *Fault Simulation, Procedure 1.* Simulate each representative fault in a G-logic partition. The procedure used shall be equivalent to the following:

Step 1: Denote by n the total number of representative faults in the G-logic partition.

Step 2: Fault simulate each representative fault. Denote by d the number of hard detections.

Step 3: Fault coverage for the G-logic partition is given by d/n.

NOTE: For example, let $n = 10,000$ (total faults in the logic model of the DUT) and suppose that $d = 9,499$ (the number of hard detections). Then the fault coverage for this logic partition is

d/n

$= 9499/10000$

$= 0.9499$

$= 94.99\%$

which can be reported as 94.99%, 94.9%, or 94%, but not as 95% (see 1.1f).

(End of Note)

3.4.2.2.2 *Fault Simulation Procedure 2.* Obtain lower bound on actual fault coverage in a G-logic partition using fixed sample size. Reference table III. The procedure used shall be equivalent to the following:

Step 1: Select a value for the penalty parameter r (r = 0.01 to 0.05). The corresponding value of n in table III is the size of the random sample of representative faults.

Step 2: Fault simulate each of the n representative faults. Denote by d the number of hard detections.

Step 3: The lower bound on the fault coverage is given by $d/n - r$.

NOTE: The penalty parameter r determines both the size of the random sample of faults and the accuracy of d/n as an estimate of the fault coverage. As the value of r increases the sample size decreases, but so does the accuracy of d/n as an estimate. Subtracting r from d/n accounts for the variance of the estimate.

For example, select $r = 0.02$. From table III, the sample size n is determined to be 1,740. Suppose that the number of hard detections d is found to be 1,679. Then the fault coverage is

$d/n - r$

$= 1679/1740 - 0.02$

$= 0.9649 - 0.02$

$= 0.9449$

$= 94.49\%$

The sample sizes given in table III are derived using the binomial distribution. Let D be the random variable denoting the number of hard detections out of n faults sampled. For a given value of r, the sample size given in table III is the smallest n, that is a multiple of 10, such that

$Pr\{D \leq n'(F + r)\} \geq 95\%$, for all n' and F such that $n' \geq n$ and $0 \leq F \leq 1 - r$

In the literature there are many approaches for statistical fault sampling. Most of the techniques are based on selecting a sample size such that the difference between the true fault coverage and the estimated fault coverage d/n does not exceed some specified value, such as ± 0.01 or ± 0.03. Such an approach results in a reported fault coverage that may be higher than the actual fault coverage. Fault Simulation Procedure 2 yields a value that, with high probability, is *lower* (by a small amount) than the actual fault coverage. *(End of Note)*

3.4.2.2.3 *Fault Simulation Procedure 3.* Accept/reject lower bound on fault coverage in a G-logic partition using fixed sample size. Reference table IV. The procedure used shall be equivalent to the following:

Step 1: Denote by F the minimum required value for fault coverage. From table IV obtain the minimum required sample size, denoted by n.

Step 2: Fault-simulate each of the n representative faults, and denote by d the number of hard detections.

Step 3: If $d < n$ (that is, any faults are undetected), then conclude that the fault coverage is less than F. Otherwise (that is, all sampled faults are detected), conclude that the fault coverage is greater than or equal to F.

NOTE: For example, suppose that the minimum required fault coverage is 95%. From table IV, the sample size n for F 95% is 59. Suppose that the number of hard detections d is found to be 50; we conclude that the fault coverage is less than 95%. On the other hand, suppose that the number of hard detections is found to be 59; this equals the number of sampled faults so we conclude that the fault coverage is greater than or equal to 95%.

Of course, if simulation of the n faults is serial and a fault is found to be undetected at any point during the process, terminate the procedure immediately and conclude that the actual fault coverage is less than F.

The sample sizes given in table IV are derived using the binomial distribution. Given that the actual fault coverage is F let D be the random variable denoting the number of hard detections out of n faults sampled. For each value of F the smallest value for n was determined such that $Pr\{D \quad n\} \quad 5\%$.

This procedure is designed so that the probability of a Type I error (that is, concluding that the actual fault coverage is greater than F, when in fact it is less than or equal to F) is less than or equal to 5%. Suppose that the minimum required level of fault coverage is 90%. This procedure is very conservative because, if the actual fault coverage is 90%, then the "hypothesis" that the actual fault coverage is greater than or equal to 90% will be rejected with probability 95%. Table IV shows that in order to have a 50% probability of accepting the hypothesis that "the actual fault coverage is greater than or equal to 90%" the actual fault coverage must be 97.6%. *(End of Note)*

3.4.3 *B-Logic.* Fault coverage shall be measured indirectly for each B-logic partition. For a given B-logic partition, the established fault coverage or test effectiveness shall be reported for that B-logic partition only if it is shown that: (a) the test vector sequence applied to the DUT applies the established test algorithm to the B-logic partition, *and* (b) the resulting critical output values from the B-logic partition are made observable at the primary outputs. Otherwise, the fault coverage for that B-logic partition shall be reported as 0%. For each B-logic partition tested in this way the established test algorithm, proof of its successful application, and the established fault coverage or test effectiveness shall be documented in the Fault Simulation Report.

3.5 *Fault Coverage Calculation.* For a given logic model for a DUT, a set of logic partitions, and a given test vector sequence, let

- m denote the number of logic partitions (as defined in 3.1.3).
- F_i denote the fault coverage of the i^{th} logic partition (measured in accordance with 3.4).
- T_i denote the transistor fraction of the i^{th} logic partition (measured in accordance with 3.1.5).

Then the overall fault coverage F for the logic model for the DUT shall be calculated as

$$F = \sum_{i=1}^{m} F_i T_i$$

where F may be stated as either a fraction or percentage.

If Fault Simulation Procedure 1 is performed for each G-logic partition in the logic model of a DUT, then the fault coverage for the logic model of a DUT shall be reported as:

> "F of all detectable equivalence classes of single, permanent, stuck-at-zero and stuck-at-one faults on the logic lines of the logic model as measured by MIL-STD-883 Procedure 5012."

If Fault Simulation Procedure 2 or 3 is performed for any G-logic partition, then the fault coverage for the logic model of a DUT shall be reported as:

> "No less than F of all detectable equivalence classes of single, permanent, stuck-at-zero and stuck-at-one faults on the logic lines of the logic model, with 95% confidence, as measured by MIL-STD-883 Procedure 5012."

The confidence level of 95% shall be identified if any Fault Simulation Procedure other than Fault Simulation Procedure 1 was performed for any G-logic partition.

4. *SUMMARY*. The following details shall be specified in the applicable procurement document:

a. Minimum required level of fault coverage and method of obtaining fault coverage.

b. If a fault sampling method is permitted, guidance on selection of the random sample of faults.

c. Guidelines, restrictions, or requirements for test algorithms for B-Logic types.

The Fault Simulation Report shall provide:

a. Statement of the overall fault coverage. If there are undetectable faults due to three-state enable signal lines, then, optionally, fault coverage based on those potential detections may be reported separately.

b. Description of logic partitions.

c. Description of test algorithms applied to B-logic. For each B-logic partition tested in this way the established test algorithm, proof of its successful application, and description of its established fault coverage or test effectiveness (including classes of faults detected) shall be documented.

NOTE: For example, if a memory test algorithm that is applied to an embedded RAM detects only stuck-at faults and some decoder faults, but not coupling or pattern-sensitivity faults, that fact shall be so stated. *(End of Note)*

d. Justification for any initial condition, other than X, for any logic line or memory element.

e. Justification for any approximations used, including estimates of fault coverages, transistor fractions, and counts of undetectable faults.

f. Description of any fault equivalencing procedure used in lieu of the procedure defined by table II.

g. Justification for declaring any fault to be undetectable.

h. In the event that the test vector sequence is formatted differently between the ATE and the fault simulator, justification that fault coverage achieved on the ATE is not lower than the reported fault coverage.

i. Justification of the use of Fault Simulation Procedure 2 or 3 rather than Fault Simulation Procedure 1.

j. When fault sampling is used, description of the method of obtaining a random sample of faults.

k. In the event that the fault simulation procedure used is not obviously equivalent to Fault Simulation Procedure 1, 2, or 3, justification as to why it yields equivalent results.

l. In the event that a test technique or design-for-testability approach is used that provides additional control or observation test points beyond those provided by the DUT's primary inputs and primary outputs (see 1.1j, *Primary inputs, Primary outputs*), justification that the stated fault coverage is valid.

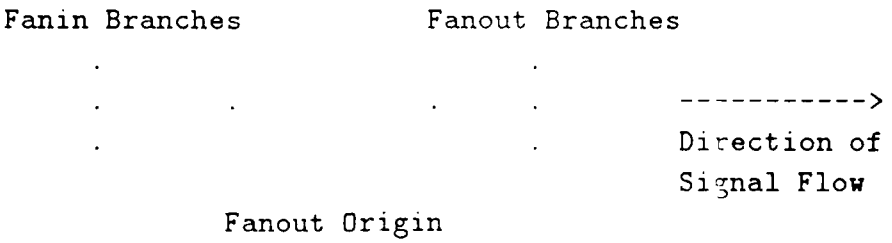

FIGURE I. Node consisting of fanin branches, a fanout origin, and fanout branches.

TABLE I. Penalty values p for LFSR signature analyzers implementing primitive polynomial of degree k.

k	p
k < 8	1.0
k = { 8..15}	0.05
k = {16..23}	0.01
k > 23	0.0

TABLE II. Representative faults for the fault equivalence classes.

Stuck-at faults	Type of logic line in logic model
s-a-1	Every input of multiple-input AND or NAND gates
s-a-0	Every input of multiple-input OR or NOR gates
s-a-0, s-a-1	Every input of multiple-input components that are not AND, OR, NAND, or NOR gates
s-a-0, s-a-1	Every logic line that is a fanout origin
s-a-0, s-a-1	Every logic line that is a primary output

Note: "s-a-0" is "stuck-at-zero" and "s-a-1" is "stuck-at-one."

TABLE III. Sample sizes used to obtain lower bound on fault coverage using Fault Simulation Procedure 2.

r	n
0.01	6860
0.015	3070
0.02	1740
0.03	790
0.04	450
0.05	290

NOTE: n is the minimum sample size required for a chosen penalty r.

TABLE IV. Sample sizes used to accept/reject lower bound on fault coverage using Fault Simulation Procedure 3.

F	n	F'
50.0%	5	87.1%
55.0%	6	89.1%
60.0%	6	89.1%
65.0%	7	90.6%
70.0%	9	92.6%
75.0%	11	93.9%
76.0%	11	93.9%
77.0%	12	94.4%
78.0%	13	94.8%
79.0%	13	94.8%
80.0%	14	95.2%
81.0%	15	95.5%
82.0%	16	95.8%
83.0%	17	96.0%
84.0%	18	96.2%
85.0%	19	96.4%
86.0%	20	96.6%
87.0%	22	96.9%
88.0%	24	97.2%
89.0%	26	97.4%
90.0%	29	97.6%
91.0%	32	97.9%
92.0%	36	98.1%
93.0%	42	98.4%
94.0%	49	98.6%
95.0%	59	98.8%
96.0%	74	99.1%
97.0%	99	99.3%
98.0%	149	99.5%
99.0%	299	99.8%

NOTE: For a given minimum required fault coverage F simulate n faults. If all faults are detected, then conclude that the actual fault coverage is greater than or equal to F. Otherwise, conclude that the actual fault coverage is less than F. The column labeled F' shows the actual fault coverage that has a 50% probability of acceptance.

MISSION

OF

ROME LABORATORY

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C3I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESC Program Offices (POs) and other ESC elements to perform effective acquisition of C3I systems. In addition, Rome Laboratory's technology supports other AFMC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.